A Deep Neural Network for Modeling Music

Pengjing Zhang, Xiaoqing Zheng,* Wenqiang Zhang, Siyan Li,
Sheng Qian, Wenqi He, Shangtong Zhang, Ziyuan Wang
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
fduzhpj@gmail.com, zhengxq@fudan.edu.cn, wqzhang@fudan.edu.cn,
leecontrolhot@gmail.com, arielqiancheng@gmail.com, maggiehwq@gmail.com,
zhangshangtong.cpp@gmail.com, ziyuan_wang@hotmail.com

ABSTRACT

We propose a convolutional neural network architecture with k-max pooling layer for semantic modeling of music. The aim of a music model is to analyze and represent the semantic content of music for purposes of classification, discovery, or clustering. The k-max pooling layer is used in the network to make it possible to pool the k most active features, capturing the semantic-rich and time-varying information about music. Our network takes an input music as a sequence of audio words, where each audio word is associated with a distributed feature vector that can be fine-tuned by backpropagating errors during the training. The architecture allows us to take advantage of the better trained audio word embeddings and the deep structures to produce more robust music representations. Experiment results with two different music collections show that our neural networks achieved the best accuracy in music genre classification comparing with three state-of-art systems.

Categories and Subject Descriptors

H.5.5 [**Sound and Music Computing**]: Methodologies and techniques, Modeling, Systems

General Terms

Algorithms, Performance

Keywords

Music classification; feature learning; neural network

1. INTRODUCTION

Over the past decade, large collections of music are increasingly available on various application platforms. Therefore, tasks such as music discovery, navigation, and organization have become progressively harder for humans without the help of automatic systems. Extensive research effort (see [5, 3, 27, 15] for reviews) has been invested in content-based music information retrieval (MIR) at the intersection of signal processing, music modeling, and machine learning. Although designed for different objectives (e.g. tagging, clustering, classification, and ranking), many MIR systems follow the same recipe with minor variations, adopting a three-step strategy: some hand-crafted features are extracted from a single or multiple frames of music audio; then, these short-term features are summarized over the entire audio to generate a music representation (or descriptor); finally, a machine learning model is trained over the summarized representations for the task of interest.

Steady improvements in the performance through perseverance and ingenuity of the MIR community indicate that this three-step recipe is effective and its optimization is not yet over. However, less attention has been paid on modeling the interaction among the segments of music, and inducing low-dimensional music representations capturing their high-level semantic information, particularly characterizing the temporal dynamics. A song normally consists of multiple sections incorporating different instrumental or vocal tones in a structured and continuous manner. The majority of musical experiences in human perception and comprehension do not live in short-term signal, but in acoustic variation over longer durations, and interplay between consecutive sections. Simply aggregating short-term features by computing statistical summaries over time generally does not produce a comprehensive and discriminative music representation because the information about temporal dynamics and order is lost, which has been proved to be very useful for music analysis and retrieval [5, 8, 11, 15, 33].

The overwhelming majority of MIR systems that follow three-step strategy operate in the paradigm of shallow architectures (with one or two levels of feature projections), which might be less efficient at representing functions that can be represented compactly by a deep architecture [1]. The advantage of the deep architectures is that high-level (normally compact) representations can be learned by multiple stacking layers, each of them combining features at lower levels in a hierarchy way. Very recently, some studies on the deep architectures (i.e. neural networks with multiple hidden layers) [10, 11, 7, 28] have shown such advantage over the shallow ones on a variety of MIR benchmarks.

In this paper, we presents a novel convolutional neural network architecture with k-max pooling layer for semantic

*Corresponding author.

ICMR'15, June 23–26, 2015, Shanghai, China.
Copyright © 2015 ACM 978-1-4503-3274-3/15/06 ...$15.00.
http://dx.doi.org/10.1145/2671188.2749367.

modeling of music. The aim of a music model is to analyze and represent the semantic content of music for purposes of classification, discovery, or clustering. An input music is transformed into a sequence of "audio words". A codebook of the audio words is constructed by clustering multiple continuous frames extracted from a collection of raw audio signals into K clusters, and each audio word is initialized by one of cluster centers learned by the clustering.

Taking a text-like music representation as input, the neural network produces local features around each audio word of the music thanks to convolutional layers, and combines these features by k-max pooling into a global feature vector or representation which can then be fed to standard affine layers. The k-max pooling operation captures the k most active features produced by the topmost convolutional layer. The pooling result is insensitive to their specific positions, but preserves the order of these features. The desirable properties of the k-max pooling layer make it possible to capture the characteristics of the temporal dynamics in music.

Given a task of interest, a relevant representation of each audio word is given by the corresponding lookup table[1] feature vector, which can be trained by backpropagating errors, starting from an initialization. Our architecture allows us to take advantage of better trained audio word representations, by moving the semantically similar audio words to be closer to each other than to those with dissimilar concepts in the vector space. Experiment results with two different music collections confirm the advantage of the trained audio word representations, which help to produce better song-level representations and improve the classification accuracy.

The remainder of this paper is organized as follows. Section 2 presents a brief overview of related work. In section 3, we introduce our neural network architecture. Experimental results are reported in Section 4. The conclusions are given in Section 5.

2. RELATED WORK

The goal of music modeling is to produce a robust representation from an observed audio signal, hopefully preserving important characteristics while discarding noise, redundancy or irrelevant information, and this problem is at the core of many tasks including classification, annotation, and retrieval. The better a representation is produced to reflect a desired semantic meaning, the simpler it is to assign or infer labels or concepts to it. Music modeling is usually conducted at two levels: *frame−level*, and *song-level* representation generations[2]. The frame-level representations are produced by extracting features from a single or few frames, and the song-level representations are constructed by aggregating the frame-level features by computing statistical summaries over the entire audio.

Most of the frame-level representations are based on traditionally well-known audio features, such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, spectral flux, linear prediction coefficients, etc., which are mainly adapted from earlier work in speech recognition. Typically, a music signal is broken into small, possibly overlapping frames (e.g. 46 ms) that are processed to feature extraction. Many systems have been proposed by using these low-level acoustic features for music classification [21, 29, 23], although they may differ in the features used or the classification models applied. The features are predominantly hand-crafted, and choosing the right features is crucial for these music classification systems. However, good feature representations are hard to come forth and more difficult to optimize. It has been shown that the features chosen by task-specific engineering can be replaced by the features learned from data in an unsupervised manner [19, 10, 28].

Unsupervised feature learning is usually conducted either by mapping the input signal into a new feature space of given *codebook* or by means of deep learning. A great deal of work has used Vector Quantization (VQ) as a tool for constructing the codebook or dictionary. A codebook is usually formed by the cluster centers of a collection of frame-level feature vectors. For instance, Riley et al compared three clustering algorithms for the codebook construction, and found that the K-means algorithm achieved comparative result with relatively less computational cost [30]. Seyerlehner et al proposed a multi-level clustering architecture to accelerate vector quantization [32]. McFee et al used a soft variant of K-means to cluster a collection of frame-level MFCC vectors [26]. Recently, many researchers have sought to employ sparse coding for the dictionary generation which produces better features than those learned with VQ methods.

Sparse coding (SC) has emerged as a strong alternative to traditional VQ approaches, and has been exploited for MIR applications [16]. Smith and Lewicki found that when the features are optimized for coding acoustic signal with sparse decomposition, they show striking similarities to time-domain cochlear filter estimates, and yield greater coding efficiency than conventional signal representations [34]. Henaff et al achieved state-of-art accuracy in music genre classification at that time by using an efficient approximated sparse coding approach with predictive sparse decomposition [12]. Yeh and Yang showed the superiority of sparsity-enforced codebook learning over traditional VQ-based or exemplar-based methods [43]. Coates and Ng found that it is more important to choose a good encoder than to choose a set of basis functions (i.e. dictionary) by systematically comparing several training and encoding schemes, including sparse coding and VQ methods [6]. They suggested that the dictionary can be generated by using fast VQ algorithms or randomly choosing exemplars from the training set.

There has been a great deal of recent interest in investigating unsupervised feature learning by deep architectures. The first successful instance is that of Convolutional Neural Network (CNN) based onset detection by [17]. Lee et al proposed a Convolutional Deep Belief Network (CDBN) that was applied to the audio spectrogram for music genre and artist classification [19]. Dieleman et al also employed the CDBN but based on engineered features (beat-aligned timbre and chroma) and pre-trained with a large data set for artist, genre, and key recognition tasks [7]. Hamel and Eck showed that the features learned by the Deep Belief Network (DBN) on Discrete Fourier Transforms (DFTs) of audio signal performed better than MFCCs for genre classification [10]. Schlüter and Osendorfer applied the mean-covariance RBM on audio spectrogram excerpts for similarity-based music classification task, and achieved close to state-of-the-art performance [31].

[1] A lookup table maps each of these audio word indices into a feature vector.

[2] MIR literature, such as [12], also mentioned a *segment-level* representation that aggregates the frame-level features by the similar methods, but whose time-scale is smaller than the song-level ones.

The frame-level features need to be summarized over time to generate the song-level representations, which allows us to map a variable-length sequence into a fixed-size feature vector that can be fed to a classifier. The song-level representations could be derived by at least three methods: *pooling*, *voting*, or *bag-of-frames*. The most straightforward way to summarize features is pooling that is to extract simple statistics such as maximum, minimum, mean or variance of the features over an excerpt of a period of time. Both the choice of the pooling operation and the temporal scale at which the pooling is applied have a great impact on the performance [11]. The voting method has been used by [10] as follows: after the classifier yields a prediction for each frame, all predictions for a song are averaged, and the prediction with the highest score is chosen as the winner.

The bag-of-frames (BoF) is another popular way to aggregate the local features for song-level representation [22, 20, 25, 42, 8]. Given a codebook, any frame-level representation can be replaced by the occurrence of codewords, and each music piece is represented as a histogram over the codebook. The BoF model is originated from the bag-of-words model, which was first proposed for document classification. Su et al investigated the influence of different keyword weighting schemes, including many different combinations of term frequency-inverse document frequency (tf-idf) measures [36]. The BoF models features as a probability distribution by simply counting the frequency of occurrence of the codewords, which is unable to characterize the temporal dynamics of high-level concepts like genre or mood, and arguably contribute to the "semantic gap" in music informatics [14].

To the best of our knowledge, the work closest to ours in terms of modeling music for classification are [33] and [43]. Shen et al proposed a Stochastic Music Concept Histogram (SMCH) scheme that represents a piece of music as the probability distribution over a set of predefined concepts [33]. Although both their and our models all take an input music as a sequence of audio words, theirs derive the music representations by estimating probabilistic association between the audio words and music concepts (e.g. genre or mood), while ours generates the compact representations of music by employing multiple network layers to learn several levels of feature extraction from the inputs. Moreover, they convert each short segment of music to a symbolic ID of audio words, whereas we assign each audio word a distributed vector representation that can be tuned during the training. Yeh and Yang investigated how to incorporate labeled data in the codebook learning process by constructing a subcodebook for each class and combing them to encode the input music [43]. In comparison, we leveraged the labeled data to obtain more robust and better music representations for the task of interest.

3. THE NETWORK ARCHITECTURE

A fairly recent trend in machine learning is to use deep architectures to discover multiple levels of features or representations from data. The popularity of deep architectures may be partially attributed to their biological plausibility: humans typically organize their thoughts with different levels of abstractions in hierarchical manner [1]. Although modeling music has been a long-standing topic for the MIR community, it is still a great challenge to discover robust representations that can capture the rich and time-varying information about music whose most expressive qualities probably relate to its structural changes across time [9]. In this paper, we introduce a convolutional neural network architecture with k-max pooling layer (KCNN) to learn music representations by extracting higher level features from acoustic signals, particularly characterizing their temporal dynamics. Before describing its components, we give our learning framework of neural network-based model in Figure 1.

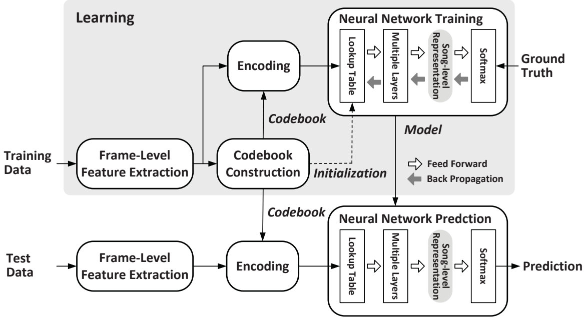


Figure 1: A learning framework of neural network-based model.

The system takes a collection of songs as training data, which are partitioned into several fix-sized frames. Some short-term features are extracted from those frames to produce their representations. The concatenations of multiple consecutive frame representations[3] are used to form segment feature vectors. To construct the codebook, the K-means algorithm is employed to cluster those vectors into groups. The audio words of the constructed codebook are used to define a mapping (an "encoding") of the input music into a new feature space. Taking a sequence of audio words as input, the first layer of our network maps each of these words into a feature vector, by a lookup table operation. A vector representation[4] of each audio word is initialized by one of cluster centers and can be fine-tuned by backpropagation with a collection of songs and their corresponding ground truth labels. The learned codebook is used to compute the encoding (i.e. a sequence of audio words) of a test song, which then is fed into the trained network for classification.

Our network architecture is shown in Figure 2. The first layer of the network extracts the frame-level features from an input music. The next convolutional layer extracts features from a window of frames, which are called "texture windows" by [38], as it corresponds to the minimum time span of music that is necessary to identify a particular music "texture". Convolutional layers are often stacked, interleaved with a non-linearity function, to extract higher level features, but just one convolutional layer is drawn in Figure 2 for clarity. The k-max pooling operation is applied after the topmost convolutional layer to pool the k most active features. It also guarantees that the input to the next layer of the network is independent of the length of the input, which allows us to apply the same subsequent layers. The following layers are standard affine layers, and the last layer is a simple softmax layer that predicts categories for the input music. The network outputs can be interpreted as a conditional probability

[3] Our initial experiments showed that using multiple consecutive frames as an input unit for learning algorithms significantly improves performance over using a single frame.

[4] It is also known as word "embeddings" in the literature of deep learning.

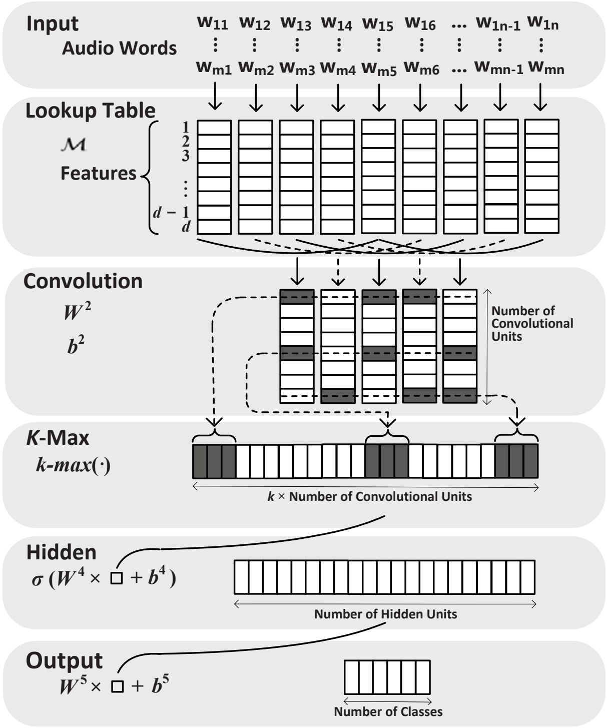


Figure 2: The neural network architecture.

by taking it to the exponential (making the score positive) and normalizing it over all possible categories.

3.1 Mapping Audio Words into Feature Vectors

The audio words are fed into the network as indices that are used by a lookup operation to transform audio words into their feature vectors. We consider a fix-sized codebook $\mathcal{D}$, whose size is equal to the number of clusters (a hyper-parameter). The vector representations are stored in a word embedding matrix $\mathcal{M} \in \mathbb{R}^{d \times |\mathcal{D}|}$, where d is the dimensionality of the vector space and $|\mathcal{D}|$ is the size of the codebook.

Formally, assume we are given a song $s_{[1:n]}$ that is first partitioned into fix-sized segments, and represented as a sequence of n audio words $s_i, 1 \leq i \leq n$. For each word $s_i \in \mathcal{D}$ that has an associated index k_i into the column of the embedding matrix, a d-dimensional feature vector representation is retrieved by the lookup table layer $\mathcal{Z}_{\mathcal{D}}(\cdot) \in \mathbb{R}^d$:

$$\mathcal{Z}_{\mathcal{D}}(s_i) = \mathcal{M} e_{k_i} \tag{1}$$

where we use a binary vector $e_{k_i} \in \mathbb{R}^{|\mathcal{D}| \times 1}$ which is zero in all positions except at the k_i-th index. The lookup table layer can be seen as a simple projection layer. If different types of features are used, we associate a lookup table with each feature, and the feature vector becomes the concatenation of the outputs of all these lookup tables. The feature vector of each word is initialized by one of the cluster centers after the codebook is constructed by vector quantization. The vector representations of audio words can be trained by backpropagation to be relevant to the task of interest. We take the audio words as the "true words" in texts, and want semantically and structurally similar words to be closer to each other than to those with dissimilar concepts in the embedding space.

3.2 Convolutional Layer

The lookup table layer extracts features for each single segment, but they are normally influenced by its surrounding segments. We assume that the features of a particular segment depend mainly on their neighboring segments, and extract these features from a fix-sized window w (a hyper-parameter) of segments. More precisely, given an input song $s_{[1:n]}$, the segment feature window produced by the lookup table layer at position s_i can be written as:

$$f_\theta^1(s_i) = \begin{pmatrix} \mathcal{Z}_{\mathcal{D}}(s_{i-w/2}) \\ \vdots \\ \mathcal{Z}_{\mathcal{D}}(s_i) \\ \vdots \\ \mathcal{Z}_{\mathcal{D}}(s_{i+w/2}) \end{pmatrix} \tag{2}$$

where f_θ^1 is a function with trainable parameters θ. The idea behind the convolution is to perform affine transformations over each segment feature window, siding over the segment $s_{[i:j]}$ from segment s_i to s_j, to obtain a matrix:

$$f_\theta^2(s_{[i:j]}) = (W^2 f_\theta^1(s_i) + b^2, \cdots, W^2 f_\theta^1(s_j) + b^2) \tag{3}$$

where the matrices $W^2 \in \mathbb{R}^{V \times (wd)}$ and $b^2 \in \mathbb{R}^V$ are parameters to be optimized during the training. A hyper-parameter V is called the *number of convolutional units*. The convolutional layer extracts local features around each window of a given segment. Multiple convolutional layers can be stacked to extract higher level features.

3.3 Max Pooling Layer

A k-max pooling operation is applied to the output of the topmost convolutional layer. The size of the outputs of Equation (3) depends on the number of segments in the input music. The local features extracted by the convolutional layers must be further combined to obtain the global feature vector with a fixed size, in order to apply the same subsequent layers independent of the length of the music.

Traditional convolutional networks often apply an average or a max operation over the time dimension of the sequence. In the context of music processing, the time here refers to the segment in a song. The average operations do not make much sense in our case, as in general not all the segments in a piece of music have equivalent influence on the meaning of its expression. In practice, the average values tend to smooth out the most active features. Instead, we use the k-max pooling approach, which forces the network to capture the k most relevant features produced by the convolutional layers. Hamel et al have shown that maximum pooling operation performs best with operations such as mean, variance, minimum, 3rd and 4th centered moments [11].

Given a number k and a sequence $\mathcal{Q} \in \mathbb{R}^p$, k-max pooling operation selects the subsequence $\mathcal{Q}_{max}^k$ of k-highest values in $\mathcal{Q}$. The selected values of $\mathcal{Q}_{max}^k$ preserve their relative order in $\mathcal{Q}$. Given a matrix output f_θ^2 by a convolutional layer, the k-max pooling layer yields a matrix:

$$f_\theta^3(f_\theta^2) = \begin{pmatrix} k\text{-max}([f_\theta^2]_{1,1} \cdots [f_\theta^2]_{1,m}) \\ \vdots \quad \vdots \quad \vdots \\ k\text{-max}([f_\theta^2]_{V,1} \cdots [f_\theta^2]_{V,m}) \end{pmatrix} \tag{4}$$

where $[f_\theta^2]_{i,j}$ is the element in the i-th row and j-th column of matrix f_θ^2, and m is the length of segments.

3.4 Label Prediction and Training

The fix-sized vector produced by concatenating each column vector of f_θ^3 is fed to the standard *Linear Layers* that successively perform affine transformations over the vector, interleaved with some non-linearity function $\sigma(\cdot)$, to extract highly non-linear features. As for a non-linear function, we choose a sigmoidal function:

$$\sigma(x) = 1/(1 + e^{-x}) \tag{5}$$

We add a simple softmax layer to the end of the networks to predict category labels for each input song. The neural networks are trained by minimizing the cross-entropy error of the softmax layer using the gradient descent algorithm. This is equivalent to minimizing the KL-divergence between the prediction distribution y and the target distribution t. We use the 1-of-K coding scheme in which the target distribution t for an instance i belonging to a class k is a binary vector with all elements zero except for the element k. The error as a function of the network parameters $\theta = (\mathcal{M}, \mathbf{W}, \mathbf{b})$ for n training instances is:

$$E(\theta) = -\frac{1}{n}\sum_{i=1}^{n} t_i \log(y_i(\theta)) + \frac{\lambda}{2}||\theta||^2 \tag{6}$$

where the coefficient λ governs the relative importance of the regularization term compared with the error.

4. EXPERIMENTS

We conducted three sets of experiments. The goal of the first one is to test several variants for the KCNN on the development set, to gain some understanding of how the choice of hyperparameters impacts the performance. We ran this set of experiments on the GTZAN[5] dataset. Ninety percent of the samples (900) were randomly chosen for training, and the rest (100) was used as development set. The second is to see how well the k-max pooling operation and the audio word representations fine-tuned by supervised learning work in improving the performance. We turned off the effect of the k-max pooling by setting the value of k to 1. We also ran a set of experiments to see the impacts of the fine-tuned audio word representations on the performance by keeping the lookup tables unchanged at the supervised training stage after they were initialized. In the third set of experiments, we compared the performance of the KCNN with the existing state-of-the-art models on two different music collections by extracting their song-level representations to train and test the SVM-based classifiers.

4.1 Datasets

We used two different datasets to evaluate the performance on music genre classification, one of most well studied problems in MIR [8, 15]. The first dataset GTZAN[6] is composed of 1,000 30-second clips covering ten genres[7], which is a balanced dataset, with 100 clips per genre [39]. The second dataset ISMIR2004Genre[8] consists of 1,458 full-length songs covering six genres[9], which is unbalanced (e.g. 320 examples for classical but 26 for jazz_blues). All songs in the two datasets were converted into a standard mono-channel and 22,050 Hz sampling rate WAV format.

Evaluation on GTZAN is conducted using a 10-fold cross validation, with the class distribution in each fold balanced like [41, 12, 43]. ISMIR2004Genre comes with predefined training and development split, so we used the development set for testing. The standard (average) classification accuracy was used to measure the performance, and all reported results have been averaged over ten runs.

4.2 Short-term audio representations

In this study, we take an input music as a sequence of audio words. To construct the codebook of audio words, raw audio signals are partitioned into fix-sized frames, and some short-term features are extracted from the frames. Many short-term audio features have been proposed in the literature. The spectrogram might be the most fundamental one that is computed by short-term Fourier transform [27]. It describes the time-varying energy across different frequency bands in a linear frequency scale of the signal. Moreover, we also consider two other representations.

- **Mel-spectrogram** that is computed by wrapping the linear-frequency scale into a nonlinear, Mel-frequency scale by triangular filters [27]. The Mel-frequency sacle is designed to approximate the frequency resolution of human ear, which is more sensitive to differences at low frequencies.
- **Spectral flatness measures (SFM) and Spectral crest factors (SCF)** are both related to the noisiness of audio signal [37]. SFM is defined as the ratio between geometric mean and arithmetic mean of the power spectrum, whereas SCF is computed as the ratio between the peak amplitude and the root-mean-square amplitude.

We used the MIR toolbox [18] to compute Mel-spectrogram, and the Marsyas [37] to extract SFM and SCF. The frame size and hop size were set to 1,024 samples and 50% of the frame size, respectively. As previously mentioned, we take multiple frames as an input unit for feature learning, and the segment feature vectors formed by the concatenations of multiple-frame representations. Hamel et al showed that simple Principal Component Analysis (PCA) whitened spectrogram can provide good performance by combining different type of temporal pooling [11], so we applied the PCA-whitening on the segment feature vectors to remove pair-wise correlation in the input data domain and, as a result, to reduce the data dimensionality. The PCA-whitened segment feature representations were used to learn the codebook of audio words by the K-means algorithm.

4.3 Competitors for performance comparison

Unsupervised feature learning typically can be conducted either by mapping the input signal into a new sparse feature

[5] Available at http://opihi.cs.uvic.ca/sound/genres.tar.gz

[6] Although the correctness of the data set has been argued in recent paper [35], it provides a benchmark on which music classification algorithm can be compared.

[7] The genre classes of GTZAN are classical, country, disco, hiphop, jazz, rock, blues, reggae, pop, and metal.

[8] Available at http://ismir2004.ismir.net/genre_contest/

[9] The genre classes of ISMIR2004Genre are classical (320), electronic (115), jazz_blues (26), metal_punk (45), rock_pop (101), and world (122).

space or by means of deep learning. We thus compared the performance of the KCNN against three state-of-art feature learning algorithms: sparse coding, autoencoder, and deep belief network.

- **Sparse coding** represents a given input as a sparse linear combination of elements in a dictionary, which is learned using the L1-penalized sparse coding formulation [24]. SC has been used for many MIR applications, including genre classification [16, 34, 12, 43]. We used the SPAMS[10] to train the dictionary and the L1-encoding to extract features from input data.
- **Autoencoder** is a neural network to learn a compressed, distributed representation for a set of data. It is trained to reconstruct its own inputs, typically for the purpose of dimensionality reduction [2]. A denoising autoencoder has been used for genre classification by [41]. In our experiments, the DeepLearnToolbox[11] was used to learn the codings of input data.
- **Deep belief network** can be viewed as a composition of restricted Boltzmann machines that are learned to probabilistically reconstruct its inputs [13]. Hamel and Eck applied the DBN to perform genre classification [10], and Schlüter and Osendorfer used the mean-covariance RBM for similarity-based music classification [31]. It is worth mentioning that RBMs and basic classical autoencoders are very similar in their functional form, although their interpretation and the procedures used for training them are quite different. We reimplemented the system of [10] by the RBMLIB[12].

To systematically compare with the above feature learning algorithms in a controlled way, all algorithms were applied on the same set of short-term features (see Section 4.2) to learn the frame-level representations. Since the learned feature vectors are generally sparse and high-dimensional, we used the max-pooling to aggregate the frame-level representations over the entire audio to produce the song-level representations. The max-pooling has been widely used [7, 28], and shown to achieve consistently higher performance on benchmark datasets over the min, mean and variance pooling operations [11].

The song-level representations learned by different algorithms were used to train the SVM-based classifiers and evaluated on the test data. We separated out the contributions of classifiers to ensure fairness of comparison. The song-level representations of the KCNN were extracted by collecting the outputs of the topmost hidden layer (see the shaded ellipses in Figure 1). The convolutional neural networks were also used to perform many MIR tasks including onset detection [17] and genre classification [7]. To compare with the classical CNN and its pretrained variant, the KCNN was turned off the effect of k-max pooling and adaptable lookup tables to approximate their results.

4.4 The choice of hyper-parameters

We tuned the hyper-parameters by trying only a few different networks. All the test results were obtained over ten runs with different random initialization for each setting of the network.

Table 1: Hyper-parameters of the KCNN.

Hyper-parameter	Value
Window size	5
Value k in k-max pooling operation	5
Number of convolutional units	500
Number of hidden units	300
Learning rate	0.02
Regularization parameter λ	0.0001

Generally, the numbers of convolutional and hidden units, provided they are large enough, have a limited impact on the generalization performance. The value k of k-max pooling layer was set to 5, which achieved a good trade-off between training speed and performance. We found that the performance improve marginally when the window size is larger than 5, but the training time increases drastically. Training could be faster with a larger learning rate, but we preferred to stick to a small one which works, rather than finding the optimal one for speed. The hyper-parameters of the network used in all the following experiments are shown in Table 1.

4.5 Results

We report the experimental results for the GTZAN in Table 2, in which our networks are indicated by "KCNN". The "ALL" in the first row indicates that all the three features were used to construct the codebook of audio words. From these numbers, a handful of trends are readily apparent. First, we note that the "full-fledged" KCNN is superior to that whose lookup tables are not modified in the supervised training stage (indicated with "ULT") by a fairly significant margin (7.68% in average). Another striking result of these experiments is the success of the k-max pooling layer, which boosts the classification accuracy about 7.23% in average. In addition, we do get surprised that it is possible to achieve better performance simply by using the nonlinear SVM as classifier instead of the softmax layer.

Table 2: Classification accuracy for the GTZAN.

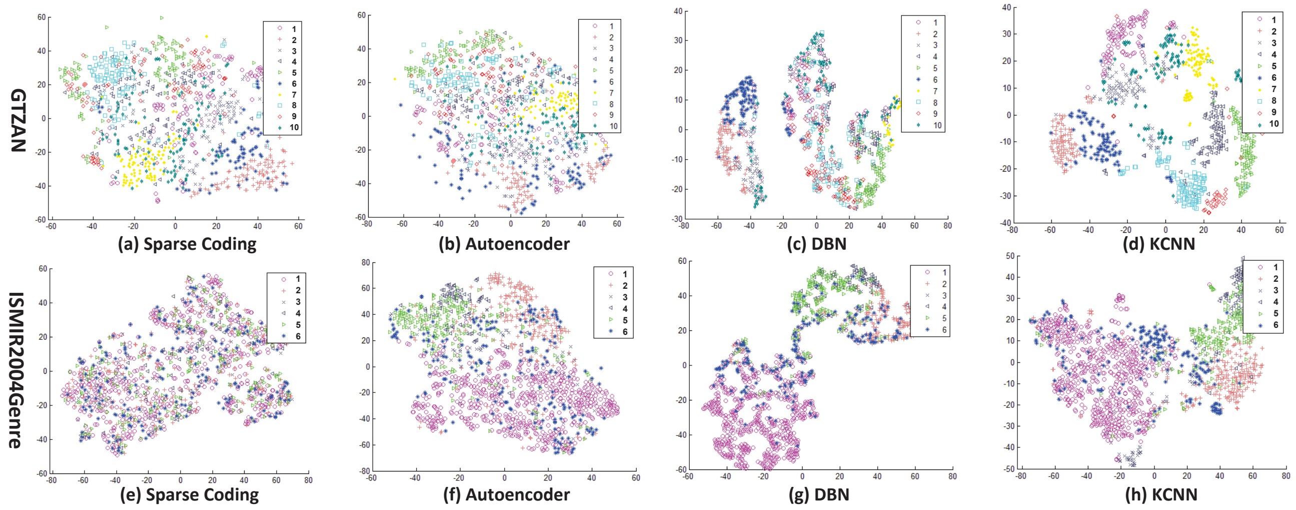
Algorithms	MEL	SFM	SCF	ALL
Sparse coding + SVM	74.30	59.00	55.60	75.10
Autoencoder + SVM	59.90	48.80	55.70	64.30
DBN + SVM	63.20	45.50	42.30	60.90
KCNN ($k = 5$) ULT	61.80	47.50	39.80	66.30
KCNN ($k = 5$)	75.30	50.90	42.30	77.60
KCNN ($k = 1$) + SVM	72.90	56.50	39.80	74.60
KCNN ($k = 5$) + SVM	80.70	59.60	48.50	83.90

The results reported in Table 2 also show that the KCNN outperforms the other unsupervised feature learning methods for music genre classification problem, highlighting the potential of the proposed network architecture for practical music classification systems. To the best of our knowledge, by far the two best results for GTZAN were achieved by Hamel and Eck [10] (84.30%), and Yeh and Yang [43] (84.70%). Although the proposed KCNN does not outperform these two prior arts, it performs competitively. The main goal of this study is to investigate how well the KCNN can produce the compact song-level representations comparing with the other unsupervised feature learning algorithms. The two best results were achieved by high-dimensional representations (taking the concatenation of the outputs of

[10] Available at http://spams-devel.gforge.inria.fr/

[11] Available at https://github.com/rasmusbergpalm/

[12] Available at https://code.google.com/p/matrbm/



(a) Sparse Coding (b) Autoencoder (c) DBN (d) KCNN

(e) Sparse Coding (f) Autoencoder (g) DBN (h) KCNN

Figure 3: Two-dimensional projections of different representations of the audio with respect to their genres.

all layers as representation [10] or using the codebook with larger size [43]) that might be inappropriate for indexing and similarity-based music recommendation purposes. Furthermore, we tried only a few different network configurations, and there are many ways (such as unsupervised pre-training, and dynamic k-max pooling) that we could improve it further. The results for the ISMIR-2004Genre is reported in Table 3, and we found the similar trends as that for the GTZAN. The results for both benchmark datasets show that the KCNN achieves consistently higher performance over the three competitors.

Table 3: Classification accuracy for the ISMIR2004-Genre.

Algorithms	MEL	SFM	SCF	ALL
Sparse coding + SVM	72.44	62.05	60.80	72.52
Autoencoder + SVM	70.03	69.88	63.55	64.61
DBN + SVM	69.50	68.45	58.30	70.08
KCNN ($k = 5$) ULT	70.03	53.92	53.57	71.08
KCNN ($k = 5$)	72.74	63.40	61.60	73.49
KCNN ($k = 1$) + SVM	69.28	63.70	69.28	71.39
KCNN ($k = 5$) + SVM	78.29	68.52	69.67	79.68

4.6 Visualization

To illustrate how well the song-level representations were learned by the KCNN, we plotted a two-dimensional projection of the representations produced by the KCNN and the other three competitors in Figure 3. We used the t-SNE algorithm [40] to perform the projection. It is worth noting that the clustering of the song-level representations generated by the KCNN is more definite than those by the sparse coding, autoencoder, and DBN for both GTZAN and ISMIR2004Genre datasets. It gives the evidence that the KCNN can derive better representations, which help to improve the accuracy of the classifiers.

5. CONCLUSION

We have described a novel convolutional neural network architecture with k-max pooling layer for semantic modeling of music. The convolutional nature of the network allows us to model the interaction among the neighboring segments of music, and summarize the features over musically significant timescales. The k-max pooling operation is designed to pool the k most relevant features produced by the topmost convolutional layer, with the order of pooled features preserved, which makes it possible to capture the characteristics of the temporal dynamics in music. More importantly, the representations of audio words initialized by vector quantization can be fine-tuned by backpropagating errors at the supervised training stage, leading to a significant increase in accuracy. Experiment results with two different music datasets show that the KCNN outperformed the three state-of-art unsupervised feature learning methods for music genre classification. For future work, we would like to test the KCNN on the other MIR tasks, such as automatic tagging and artist recognition. It also would be interesting to see how well our network scales to much larger music databases, by applying it to a larger dataset, such as the Million Song dataset [4].

6. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by a grant from Shanghai Municipal Natural Science Foundation (No. 13ZR1403800).

7. REFERENCES

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[2] Y. Bengio and Y. LeCun. *Scaling learning algorithms towards AI. In Large Scale Kernel Machines, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (eds)*. MIT Press, 2007.

[3] T. Bertin-Mahieux, D. Eck, and M. Mandel. *Automatic tagging of audio: The state-of-art. In Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[4] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *International Society for Music Information Retrieval (ISMIR'11)*, 2011.

[5] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *IEEE*, 96(4):668–696, 2008.

[6] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning (ICML'11)*, 2011.

[7] S. Dielemann, P. Brakel, and B. Schrauwen. Audio-based music classification with a pretrained convolutional network. In *International Society for Music Information Retrieval (ISMIR'11)*, 2011.

[8] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Transaction on Multimedia*, 13(99):303–319, 2011.

[9] P. Gomez and B. Danuser. Relations between musical structure and psychophysiological measures of emotion. *Emotion*, 7(2):377–87, 2007.

[10] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *International Society for Music Infromation Retrieval (ISMIR'10)*, 2010.

[11] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *International Society for Music Infromation Retrieval (ISMIR'11)*, 2011.

[12] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCuu. Unsupervised learning of sparse features for scalable audio classification. In *International Society for Music Information Retrieval (ISMIR'11)*, 2011.

[13] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[14] E. J. Humphrey, J. P. Bello, and Y. LeCun. Moving beyond feature design: deep architectures and automatic feature learning in music informatics. In *International Society for Music Information Retrieval (ISMIR'12)*, 2012.

[15] M. Kaminskas and F. Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6:89–119, 2012.

[16] P. D. C. Kereliuk. Sparse atomic modeling of audio: A review. In *International Conference on Digital Audio Effects (DAFx'11)*, 2011.

[17] A. Lacoste and D. Eck. A supervised classification algorithm for note onset detection. *EURASIP Journal on Applied Signal Processing*, (1):1–14, 2007.

[18] O. Lartillot and P. Toiviainen. A matlab toolbox for musical feature extraction from audio. In *International Conference on Digital Audio Effects (DAFx'07)*, 2007.

[19] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Neural Information Processing Systems (NIPS'09)*, 2009.

[20] M. Levy and M. Sandler. Music information retrieval using social tags and audio. *IEEE Transactions on Multimedia*, 11:383–395, 2009.

[21] G. Li and A. A. Khokhar. Content-based indexing and retrieval of audio data using wavelets. In *International Conference on Multimedia and Expo (ICME'00)*, 2000.

[22] L. Lu and A. Hanjalic. Audio keywords discovery for text-like audio content analysis and retrieval. *IEEE Transactions on Multimedia*, 10(1):74–85, 2008.

[23] L. Lu, H. Zhang, and S. Z. Li. Content-based audio classification and segmentation by using support vector machines. *Multimedia Systems*, 8(6):482–492, 2003.

[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML'09)*, 2009.

[25] M. I. Mandel, D. Eck, and Y. Bengio. Learning tags that vary within a song. In *International Society for Music Information Retrieval (ISMIR'10)*, 2010.

[26] B. McFee, L. Barrington, and G. Lanckriet. Learning content similarity for similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:2207–2218, 2012.

[27] M. Muller, D. P. W. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1088–1110, 2011.

[28] J. Nam, J. Herrera, M. Slaney, and J. Smith. Learning sparse feature representations for music annotation and retrieval. In *International Society for Music Information Retrieval (ISMIR'12)*, 2012.

[29] U. Nam and J. Berger. Addressing the same but different—different but similar problem in automatic music classification. In *International Symposium on Music Information Retrieval*, 2001.

[30] M. Riley, E. Heinen, and J. Ghosh. A text retrieval approach to content-based audio retrieval. In *International Society for Music Information Retrieval (ISMIR'08)*, 2008.

[31] J. Schlẗer and C. Osendorfer. Music similarity estimation with the mean-covariance restricted boltzmann machine. In *International Conference on Machine Learning and Applications (ICMLA'11)*, 2011.

[32] K. Seyerlehner, G. Widmer, and P. Knees. Frame level audio similarity: A codebook approach. In *International Conference on Digital Audio Effects (DAFx'08)*, 2008.

[33] J. Shen, H. Pang, M. Wang, and S. Yan. Modeling concept dynamics for large scale music search. In *Special Interest Group on Information Retrieval (SIGIR'12)*, 2012.

[34] E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439(7079):978–982, 2006.

[35] B. L. Sturm. An analysis of the gtzan music genre dataset. In *ACM Workshop Music Information Retrieval with User-Centered and Multimodal Strategies*, 2012.

[36] L. Su, C.-C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang. A systematic evaluation of the bag-of-frames representation for music information retrieval. *IEEE Transactions on Multimedia*, 16(5):1188–1200, 2014.

[37] G. Tzanetakis. *Marsyas: a case study in implementing Music Information Retrieval Systems. In Intelligent Music Information Systems: Tools and Methodologies, Shen, Shepherd, Cui, and Liu (eds)*. IGI Publishing, 2008.

[38] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:93–302, 2002.

[39] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *International Society for Music Information Retrieval (ISMIR'01)*, 2001.

[40] L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[42] J.-C. Wang, H.-S. Lee, H.-M. Wang, and S.-K. Jeng. Learning the similarity of audio music in bag-of-frames representation from tagged music data. In *International Society for Music Information Retrieval (ISMIR'11)*, 2011.

[43] C.-C. M. Yeh and Y.-H. Yang. Supervised dictionary learning for music genre classification. In *International Conference on Multimedia Retrieval (ICMR'12)*, 2012.